APPENDIX

```
;---- DEFINE FILES USED ------------------------------------------------------
;  Proper listing of all files being use during compiling of the source code.
;==== BEGIN DEFINE ============================================================
   DEFINE  TRI_CODE
         INCLUDE "TRICODE_DEF.INC"
   SEGMENT TRI_CODE
;==== END DEFINE ==============================================================


;---- IRQ VECTOR DEFINITION --------------------------------------------------
; Define the subroutine that will be called upon interupt (if enabled).
; IRQ3 (P30) will be the only one that is enabled for this project.
; This is the RADIO input interupt.
;==== BEGIN IRQ VECTOR ========================================================
   ORG     %00                          ;Top of program memmory.
      .WORD   RADIO_INTERUPT           ;IRQ0, P32 input from radio input circuit.
      .WORD   IRQ1_VECTOR              ;IRQ1, P33
      .WORD   IRQ2_VECTOR              ;IRQ2 vector, P31
      .WORD   IRQ3_VECTOR              ;IRQ3, P30
      .WORD   T0_INTERUPT              ;IRQ4, T0 interupt.
      .WORD   T1_INTERUPT              ;IRQ5, T1 interupt.
;==== END IRQ VECTOR ==========================================================


;==== BEGIN MAIN CODE =========================================================
MAIN:
```

```
ORG     %0C                             ;Beginning address of ROM code

;Initializing WDT Section==========;Must be done during the first 60 instruction cycles
WDT                                 ;Enable WDT.
srp     #0FH                            ;Select Working Reg Group0, Expand Reg GroupF
ld      WDTMR,#00001111B                ;Int RC osc. -80 msec,WDT on during HALT or STOP.
clr     RP                              ;Return Register Pointer to Bank0 (standard)
;Initializing WDT Section==========;

DI                                  ;Disable all interupt.

;Initializing I/O port section======;
ld    PCON,#pcon_ini                ;Initial value of PCON See TriCodeRx.inc for details.
ld    P01M,#p01m_ini                ;
ld    P2M,#p2m_ini                  ;
ld    P2M_COPY,#p2m_ini             ;Mirroring P2M
ld    P3M,#p3m_ini                  ;
clr   P0                            ;
clr   P2                            ;
clr   P3                            ;
;Initializing I/O port section======;

;Initializing INTERUPT section======;
ld    IPR,#ipr_ini                  ;Priority IRQ0,2,4,1,5,3.
ld    IRQ,#irq_ini                  ;Clear out all IRQ initially.
ld    IMR,#imr_ini                  ;Disable all interupts initially.
;Initializing INTERUPT section======;

;Intializing TIMER section =========;
ld    TMR,#tmr_ini                  ;Initial values for timer control registers.
ld    PRE0,#pre0_100usec                ;
ld    T0,#t0_1msec                  ;
clr     PRE1                        ;
clr     T1                          ;
RESET_MULTIPLIER                    ;clear T0 multiplier.
;Intializing TIMER section =========;

    jp    START                     ;Goto Start of program.
;===== END MAIN CODE ================================================================


;*********************************************************************************
;*              BEGIN INTERUPT SERVICE ROUTINES SECTION                          *
;*********************************************************************************
;===== RADIO INTERUPT SERVICE ROUTINE ===============================================
;This routine is the 'brain' of determining the validity of the signal received.
;Input :
;  1. Receiver mode: A , B  or C  (CONTROL_INPUTS)
;  2. T0 properties: T0_MULTIPLIER
;Output:
;  1. Data bits value for valid signal.
;  2. Reset control bits to indicate invalid data.
;===== BEGIN RADIO INTERUPT SERVICE ROUTINE =========================================
RADIO_INTERUPT:
   WDT
   push  FLAGS                      ;Save FLAGS register for normal operation.
   ld    T0_VALUE,T0                ;Save T0 value for calculation later.
   RELOAD_T0                        ;Reload the initial value for T0.
   TEN_MSEC_NOT_PASSED              ;
   cp    T0_MULTIPLIER,#012H        ;Check if at least 18 msec has elapsed.
   jp    ugt,FIRST_EDGE             ;If so this is the first edge of the first pulse.
   tm    DATA_FLAGS,#00000010B      ;Test if first edge already detected.
   jp    z,EXIT_RADIO_INT           ;If not then just exit.
NEXT_EDGE:
   RESET_MULTIPLIER                 ;Reset the 1msec multiplier for T0.
   tm    P3,#00000100B              ;Check the present input state of radio's input.
   jp    nz,LOW_PULSE_WIDTH         ;If 'HI' then current pulse width is low.
```

```
   ;===== VERIFYING ONTIME PULSE WIDTH ==========================================
HI_PULSE_WIDTH:
   tm    CONTROL_INPUTS,#00000100B  ;Test for " B " mode.
   jp    nz,LINEAR_ON_TIME_CHECK    ;If it is then goto check its 'on time' width.
      ;==== MULTI/STANLEY PULSE CHECK ==========================================
MULTI_ON_TIME_CHECK:                ;else check " C "/" A "  ontime width.
   cp    T0_MULTIPLIER,#01H         ;Check the 1msec multiplier.
   jp    ugt,NOT_VALID_DATA_BIT     ;if greater than or equal 2 msec then not good.
   jp    z,ATLEAST_1MSEC            ;If equal to 1 then check for logic '1' pulse.
         ;==== LOGIC '0' CHECK ================================================
   cp    T0_VALUE,#07H              ;Is it less than or equal .3 msec.
   jp    ugt,NOT_VALID_DATA_BIT     ;If so then not good data.
   cp    T0_VALUE,#05H              ;Is it more than or equal .6 msec.
   jp    ult,NOT_VALID_DATA_BIT     ;If so then data is no good.
   LOGIC_0                          ;.3 <= WIDTH <= .6 therefor logic '0'.
   jp    GOOD_ON_TIME_MEASURED      ;Goto good on time measured.
         ;==== LOGIC '0' CHECK ================================================

         ;==== LOGIC '1' CHECK ================================================
ATLEAST_1MSEC:
   cp    T0_VALUE,#08H              ;Is it less than or equal 1.2 msec.
   jp    ugt,NOT_VALID_DATA_BIT     ;If so then data is no good.
   cp    T0_VALUE,#04H              ;Is it greater than or equal 1.7 msec.
   jp    ult,NOT_VALID_DATA_BIT     ;if so data is no good.
   LOGIC_1                          ;1.2 <= WIDTH <= 1.7 therefor logic '0'.
   jp    GOOD_ON_TIME_MEASURED      ;Goto good on time measured.
         ;==== LOGIC '1' CHECK ================================================
      ;==== MULTI/STANLEY PULSE CHECK ==========================================
LINEAR_ON_TIME_CHECK:

   ;===== VERIFYING ONTIME PULSE WIDTH ==========================================

   ;===== VERIFYING OFFTIME PULSE WIDTH =========================================
LOW_PULSE_WIDTH:
   cp    BIT_COUNTER,#0AH           ;Is bit counter at 10.
   jp    z,NOT_VALID_DATA_BIT       ;If it reaches 10 then there is something wrong
   tm    DATA_FLAGS,#00000001B      ;Test for a good ontime was measured previously.
   jp    z,NOT_VALID_DATA_BIT       ;If it's not then don't bother to check off time.
   tm    CONTROL_INPUTS,#00000100B  ;Test for " B "  mode.
   jp    nz,LINEAR_OFF_TIME_CHECK   ;If it is then goto check its 'off time' width.
MULTI_OFF_TIME_CHECK:



LINEAR_OFF_TIME_CHECK:
   ;===== VERIFYING OFFTIME PULSE WIDTH ==========================================

   ;===== GOOD ON TIME MEASURED SECTION ==========================================
GOOD_ON_TIME_MEASURED:
   or    DATA_FLAGS,#00000001B      ;Set good ontime pulse flag bit.
   jp    EXIT_RADIO_INT
   ;===== GOOD ON TIME MEASURED SECTION ==========================================

VALID_DATA_BIT:
   jp    EXIT_RADIO_INT

NOT_VALID_DATA_BIT:
   and   DATA_FLAGS,#11111100B      ;Clear good 'ontime' and the 'first edge' flag bit.
EXIT_RADIO_INT:
   pop     FLAGS
   iret

   ;==== FIRST EDGE INTERUPT SECTION ============================================
FIRST_EDGE:
   RESET_MULTIPLIER                 ;Reset T0 multiplier to zero.
   RESET_BIT_COUNTER                ;
```

```
   RESET_BIT_1_POSITION             ;
   RESET_BIT_0_POSITION             ;
   or    DATA_FLAGS,#00000010B      ;Set bit1 for first edge detected.
   jp    EXIT_RADIO_INT
   ;==== FIRST EDGE INTERUPT SECTION ==========================================
;===== END RADIO INTERUPT SERVICE ROUTINE ======================================


;==== IRQ1_VECTOR INTERUPT SERVICE ROUTINE =====================================
;==== BEGIN IRQ1_VECTOR INTERUPT SERVICE ROUTINE ===============================
IRQ1_VECTOR
   iret
;==== END IRQ1_VECTOR INTERUPT SERVICE ROUTINE =================================

;==== IRQ2_VECTOR INTERUPT SERVICE ROUTINE =====================================
;==== BEGIN IRQ2_VECTOR INTERUPT SERVICE ROUTINE ===============================
IRQ2_VECTOR
   iret
;==== END IRQ2_VECTOR INTERUPT SERVICE ROUTINE =================================

;==== IRQ3_VECTOR INTERUPT SERVICE ROUTINE =====================================
;==== BEGIN IRQ3_VECTOR INTERUPT SERVICE ROUTINE ===============================
IRQ3_VECTOR
   iret
;==== END IRQ3_VECTOR INTERUPT SERVICE ROUTINE =================================
;==== INTERUPT SERVICE ROUTINES SECTION ========================================

;==== T0_VECTOR INTERUPT SERVICE ROUTINE =======================================
;Purpose:
;  This interupt capture T0 timer. It is started by the first state change in Radio Int.
;  at pin32 (IRQ0). It is continous counting. It should time out (get here) every 1 msec
;  The T0_MULTIPLIER is then incremented, therefore this register value can be considered
;  as the msec value.The initial value for T0 is 10 (decimal), therefor each count in T0
;  is an equivalent of .1 msec. Example: if T0 = 8 the .2 msec has elapsed.
;Input:
;  T0 reaches end of count.
;Output:
;  T0_MULTIPLIER: Store the # of T0 timeout(msec) without Radio interupt being detected.
;==== BEGIN T0_VECTOR INTERUPT SERVICE ROUTINE =================================
T0_INTERUPT
   WDT
   push  FLAGS                  ;Save flags
   inc   T0_MULTIPLIER          ;Increment the xmsec counter.
   cp    T0_MULTIPLIER,#0AH     ;Test if 10 msec has elapsed.
   jp    z,NO_PULSE_FOR_10MSEC;If so set the proper bit.
   cp    T0_MULTIPLIER,#023H    ;If 35 msec (23H) has passed then stop counting.
   jp    z,STOP_COUNTING        ;If no signal for 35 msec stop the timer.
   jp    EXIT_T0_INTERUPT       ;
NO_PULSE_FOR_10MSEC:
   TEN_MSEC_PASSED              ;10 msechas passed since last pulse.
STOP_COUNTING:
   DISABLE_T0_COUNT             ;Stop T0 count.
   LD    T0_MULTIPLIER,#0FFH    ;Set all bits for overcount.(no signal for 35msec)
EXIT_T0_INTERUPT:
   pop   FLAGS                  ;restore flags value.
   iret
;==== END T0_VECTOR INTERUPT SERVICE ROUTINE ===================================

;==== T1_VECTOR INTERUPT SERVICE ROUTINE =======================================
;==== BEGIN T1_VECTOR INTERUPT SERVICE ROUTINE =================================
T1_INTERUPT
   WDT
   iret
;==== END T1_VECTOR INTERUPT SERVICE ROUTINE ===================================
;*******************************************************************************
;*              END INTERUPT SERVICE ROUTINES SECTION                          *
;*******************************************************************************
```

```
;==== READ P0 INPUTS ================================================================
;Purpose:
;  1. Read P0 inputs with debounce.Inputs must be the same for 10 consecutive read cycles
;     to be a valid input. If no valid reading can be obtained the WDT will reset the uC.
;General Algorithym:
;     P2 I/O controls which set of P0 input is being read so it's must be know prior to
;  calling this routine. The entire P0 is read minmum of 10 times, each time the previous
;  value is compared to the current value. If they are not the same (noises) then the
;  counter is reset and the process repeat. 10 consecutive identical reading is required for
;  a valid input for P0. If such condition can not be achieved then the WDT should time out
;  and reset the micro. This would avoid a infinite loop.
;Input:
;  1. P2M I/O controls: must be pre-determine by the calling routine.
;Output:
;  1. P0_INPUT: All the content of P0 is stored in P0_INPUT.
;Subroutine called:
;  1. None.
;==== BEGIN READ P0 INPUTS ==========================================================
READ_P0:
   WDT
   ld    DEBOUNCE_COUNTER,#debounce_value;load initial value for debounce counter (10)
   ld    OLD_INPUT,P0                ;Read P0 and store it as previous input.
   WDT                               ;
DEBOUNCE_P0:
   ld    NEW_INPUT,P0                ;Read P0 and store as current input.
   cp    OLD_INPUT,NEW_INPUT         ;Compare previous and current input.
   jp    z,P0_SAME                   ;If the same then goto P0_SAME.
P0_NOT_SAME:
   ld    DEBOUNCE_COUNTER,#debounce_value;reload initial value for debounce counter
   jp    SAVE_NEW_INPUT              ;
P0_SAME:
   dec   DEBOUNCE_COUNTER            ;decrement debounce counter.
   jp    z,EXIT_READ_P0              ;If zero then exit
SAVE_NEW_INPUT:
   ld    OLD_INPUT,NEW_INPUT         ;Save current input.
   jp    DEBOUNCE_P0                 ;Re-read P0 again.

EXIT_READ_P0:
   ld    P0_INPUT,NEW_INPUT          ;Store new input in P0_INPUT register.
   ret                               ;EXIT
;==== BEGIN READ P0 INPUTS ==========================================================


;==== SET RF CHANNEL ================================================================
;Purpose:
;  Set P25 hi or low according to the mode slide switch input:
;       C              = CHANNEL 1 (300 MHz) => P25 = OUTPUT/LO
;       B  /.  A       = CHANNEL 2 (310 MHz) => P25 = INPUT/HI Z.
;General Algorithm:
;     Use the input from CONTROL_INPUTS register to determine the out put of P25.
;Input:
;     CONTROL_INPUTS
;Output:
;     P25's state. No other register is affected.
;==== BEGIN SET RF CHANNEL ==========================================================
SET_RF_CHANNEL:
   WDT
   tm    CONTROL_INPUTS,#00000010B   ;Test for '. C ' input.
   jp    nz,TUNE_TO_300MHz           ;Goto and tune to 300MHz.
TUNE_TO_310MHz:
   or    P2M_COPY,#00100000B         ;Set P25 to input/Hi Z.
   ld    P2M,P2M_COPY                ;
   ret                               ;
TUNE_TO_300MHz:
```

```
   and   P2M_COPY,#11011111B        ;Set P25 to output
   ld    P2M,P2M_COPY               ;
   NOP                              ;Waste 1-cycle for change state.
   or    P2,#00100000B              ;Set out put to lo/GND.
   ret                              ;
;===== END SET RF CHANNEL ==========================================================

;===== READ CONTROL INPUTS ==========================================================
;Purpose:
;  1. Read S3 switches to determine  B  , C  or  A  mode of operation.
;  2. Read P2 and P4 to determine constant pressure or momentary operation for both channel.
;General Algorithm:
;     Set P24 pin to output and set it HI (5V). Read P0 to determine the slide switch position
;  (  B  ,  C  ,  A  ) and the switch mode (constant pressure or momentary). After
;  complete the read reset the output P24 to LO then change it to an input pin (high
;  impedance). The result is stored in CONTROL_INPUT register for future use. This register
;  will not be changed until the next read cycle or this routine is being called. The value of
;  P2M is not being changed. It should be the same after this routine is exited.
;Input:
;  1. None
;Output:
;  2. CONTROL_INPUTS register bits 0,1,2,5,6 will change accordingly other remain undisturb.
;Subroutine called:
;  1. READ_P0.
;===== BEGIN READ CONTROL INPUTS ====================================================
READ_CONTROL_INPUTS:
   WDT
   and   P2M_COPY,#11101111B  ;Using P2M_COPY as intermediate/P2M is write only.
   ld    P2M,P2M_COPY         ;Change only P24 to output undisturb the rest.
   NOP                        ;Wait one cycle before put data on output pin.
   or    P2,#00010000B        ;Set P24 to Hi (Mode select switch).
   call  READ_P0              ;Read P0 with debounce.
   and   P0_INPUT,#01100111B  ;Mask out all other except for mode switches inputs.
   ld    CONTROL_INPUTS,P0_INPUT ;Store mode switch input to the proper bit location.
   and   P2,#11101111B        ;Shut off P24 output.
   NOP                        ;Wait 1 cycle before I/O change.
   or    P2M_COPY,#00010000B  ;Set P24 to high impedance input.
   ld    P2M,P2M_COPY         ;using P2M_COPY as intermediate register.
   ret                        ;
;===== END READ CONTROLS INPUTS =====================================================

;===== READ DIP SWITCH ==============================================================
;Purpose:
;  1. Read DIP switch input to determine the unique code.
;General Algorithm:
;  1. Determine the channel using CONTROL_INPUT registers bit 0,1,2. It bit 1 is high then
;     it's channel 1 (300Mhz) otherwise it's channel 2.
;  2. Once the channel is determined either S1 or S2 will be read as follow:
;        * DIP switch 1 thru 5 is read and store in DIP_SW_HIBYTE [4-0]. Example: DIP1's
;          value (D1) is stored in bit4 location of register DIP_SW_HIBYTE.
;        * DIP switch 6 thru 10 is read and store in DIP_SW_HIBYTE [4-0].
;  3. Left Rotate DIP_SW_LOWBYTE 3 times to bring D6 to bit7.
;  4. Left Rotate DIP_SW_LOWBYTE through Carry 'C' then left rotate DIP_SW_HIBYTE. Do this
;     3 times.
;  5. The final format of the register should be:
;        DIP_SW_HIBYTE: D1 D2 D3 D4 D5 D6 D7 D8
;        DIP_SW_LOWBYTE:D9 D10 0 0 0 0 0 0
;
;Input:
;  1. CONTROL_INPUTS: to determine which set of DIP switch to read from.
;Output:
;  1. DIP_SW_LOWBYTE, DIP_SW_HIBYTE: Store the ID code read from DIP switches
;Subroutine called:
;  1. READ_P0.
;===== BEGIN READ DIP SWITCH ========================================================
READ_DIP_SWITCH:
```

```
   WDT
   tm    CONTROL_INPUTS,#00000010B  ;Test for C     mode select.
   jp    nz,READ_S1                 ;If it is then read S1 inputs
                                    ;Otherwise read S2 (Linear or Stanley)
   ;----- READ DIP SWITCH (S2) SECTION ----------------------------------------------------
                          ;Otherwise read S2 inputs.
READ_S2:
   ;----- READ DIP1-5-----------------------
   and   P2M_COPY,#11110111B        ;Use P2M_COPY to change P23 to output pin.
   ld    P2M,P2M_COPY               ;Write it to P2M to change P23 to output.
   NOP                              ;The rest of P2M remain unchanged.
   or    P2,#dip_2_1_select         ;Turn P23 on (Hi) to select dip1-5 to be read
   call  READ_P0                    ;Read P0
   and   P2,#11110111B              ;Reset P23 to lo.
   or    P2M_COPY,#00001000B        ;Change P23 back to input pin (Hi Z).
   ld    P2M,P2M_COPY               ;
   ld    DIP_SW_HIBYTE,P0_INPUT     ;Save P0_INPUT to DIP_SW_HIBYTE.
   and   DIP_SW_HIBYTE,#00011111B   ;Mask out non-DIP switch input
   ;----- READ DIP1-5-----------------------

   ;----- READ DIP6-10----------------------
   and   P2M_COPY,#11111011B        ;Use P2M_COPY to change P22 to output pin.
   ld    P2M,P2M_COPY               ;Write it to P2M to change P22 to output.
   NOP                              ;The rest of P2M remain unchanged.
   or    P2,#dip_2_2_select         ;Turn P22 on (Hi) to select dip6-10 to be read
   call  READ_P0                    ;Read P0
   and   P2,#11111011B              ;Reset P22 to lo.
   or    P2M_COPY,#00000100B        ;Change P22 back to input pin (Hi Z).
   ld    P2M,P2M_COPY               ;
   ld    DIP_SW_LOWBYTE,P0_INPUT    ;Save P0_INPUT to DIP_SW_LOWBYTE.
   and      DIP_SW_LOWBYTE,#00011111B  ;Mask out non-DIP switch input
   ;----- READ DIP6-10----------------------

   ld    ROTATE_COUNTER,#03H        ;Load number of rotations desired (3)
ROTATE_LOWBYTE_S2:
   rlc   DIP_SW_LOWBYTE             ;left rotate lower byte 3 times to bring D6
   dec   ROTATE_COUNTER             ;thru D10 to the left most bits.
   jp    nz,ROTATE_LOWBYTE_S2       ;

   ld    ROTATE_COUNTER,#03H           ;Load number of rotations desired (3)
ROTATE_BOTH_BYTE_S2:
   rlc   DIP_SW_LOWBYTE             ;left rotate lower byte 3 times through carry
   rlc   DIP_SW_HIBYTE              ;left rotate hi byte 3 times through carry.
   dec   ROTATE_COUNTER             ;
   jp    nz,ROTATE_BOTH_BYTE_S2     ;
   jp    EXIT_READ_DIP_SWITCH       ;Finalize and exit subroutine.
   ;----- READ DIP SWITCH (S2) SECTION ----------------------------------------------------

   ;----- READ DIP SWITCH (S1) ------------------------------------------------------------
READ_S1:
   ;----- READ DIP1-5-----------------------
   and   P2M_COPY,#11111101B        ;Use P2M_COPY to change P21 to output pin.
   ld    P2M,P2M_COPY               ;Write it to P2M to change P21 to output.
   NOP                              ;The rest of P2M remain unchanged.
   or    P2,#dip_1_1_select         ;Turn P21 on (Hi) to select dip1-5 to be read
   call  READ_P0                    ;Read P0
   and   P2,#11111101B              ;Reset P21 to lo.
   or    P2M_COPY,#00000010B        ;Change P21 back to input pin (Hi Z).
   ld    P2M,P2M_COPY               ;
   ld    DIP_SW_HIBYTE,P0_INPUT     ;Save P0_INPUT to DIP_SW_HIBYTE.
   and   DIP_SW_HIBYTE,#00011111B   ;Mask out non-DIP switch input
   ;----- READ DIP1-5-----------------------

   ;----- READ DIP6-10----------------------
   and   P2M_COPY,#11111110B        ;Use P2M_COPY to change P20 to output pin.
   ld    P2M,P2M_COPY               ;Write it to P2M to change P20 to output.
   NOP                              ;The rest of P2M remain unchanged.
```

```
   or    P2,#dip_1_2_select        ;Turn P20 on (Hi) to select dip6-10 to be read
   call  READ_P0                   ;Read P0
   and   P2,#11111110B             ;Reset P20 to lo.
   or    P2M_COPY,#00000001B       ;Change P20 back to input pin (Hi Z).
   ld    P2M,P2M_COPY              ;
   ld    DIP_SW_LOWBYTE,P0_INPUT   ;Save P0_INPUT to DIP_SW_LOWBYTE.
   and   DIP_SW_LOWBYTE,#00011111B ;Mask out non-DIP switch input
   ;----- READ DIP6-10-----------------------

   ld    ROTATE_COUNTER,#03H       ;Load number of rotations desired (3)
ROTATE_LOWBYTE_S1:
   rlc   DIP_SW_LOWBYTE            ;left rotate lower byte 3 times to bring D6
   dec   ROTATE_COUNTER            ;thru D10 to the left most bits.
   jp    nz,ROTATE_LOWBYTE_S1      ;

   ld    ROTATE_COUNTER,#03H          ;Load number of rotations desired (3)
ROTATE_BOTH_BYTE_S1:
   rlc   DIP_SW_LOWBYTE            ;left rotate lower byte 3 times through carry
   rlc   DIP_SW_HIBYTE             ;left rotate hi byte 3 times through carry.
   dec   ROTATE_COUNTER            ;
   jp    nz,ROTATE_BOTH_BYTE_S1    ;
   ;----- READ DIP SWITCH (S1) ----------------------------------------------------------

   ;----- FINALIZING AND EXIT READ DIP SWITCH ----------------------------------------
EXIT_READ_DIP_SWITCH:
   tm    CONTROL_INPUTS,#00000001B ;Test for "B"  mode select.
   jp    nz,ZERO_LOWBYTE           ;If so goto Zeroing out low byte.
CLEAR_LAST_6_BIT:
   and   DIP_SW_LOWBYTE,#11000000B ;Clear out the last 6 bits.
   ret                             ;then exit routine.
ZERO_LOWBYTE:
   clr   DIP_SW_LOWBYTE            ;Clear out lower byte use hi byte only.
   ret                             ;then exit routine.
   ;----- FINALIZING AND EXIT READ DIP SWITCH ----------------------------------------

;==== END READ DIP SWITCH ==========================================================


;==== BEGIN START OF PROGRAM =======================================================
START:
   call  READ_CONTROL_INPUTS       ;Obtain control parameters.
   call  SET_RF_CHANNEL            ;Set P25 accordingly.
   call  READ_DIP_SWITCH           ;Obtain ID code from DIP switch.
   or    IMR,#00000001B            ;Enable IRQ0 (Radio interupt).
   EI                              ;Enable all interupts.
PROGRAM_LOOP:
   call  READ_CONTROL_INPUTS       ;Obtain control parameters.
   call  SET_RF_CHANNEL            ;Set P25 accordingly.
   call  READ_DIP_SWITCH           ;Obtain ID code from DIP switch.
   NOP                             ;
   JP    PROGRAM_LOOP              ;
;==== END START OF PROGRAM =========================================================
END
```

```
stack_top   equ   07Fh                ;Assigning top of stack at R127.
;-------------------------------------------------------------------------------

;---- INITIALIZE I/O PORTS LABELS ----------------------------------------------
;  Labels/variables related to I/O ports operation.
;-------------------------------------------------------------------------------

debounce_value    equ   0Ah                 ;maximum # for debounce counter.
dip_1_1_select    equ   00000010B           ;Use to turn on P21, shut every thing else off
dip_1_2_select    equ   00000001B           ;Use to turn on P20, shut every thing else off
dip_2_1_select    equ   00001000B           ;Use to turn on P23, shut every thing else off
dip_2_2_select    equ   00000100B           ;Use to turn on P22, shut every thing else off
dip_sw_read       equ   11110000B           ;Use to set P2M to read dip switch

p01m_ini          equ   01000101B           ;P04-P07 = input.
                                            ;Normal external memory (N/A)
                                            ;No P1 for Z86E31.
                                            ;Internal stack only.
                                            ;P03-P00 = input.
p2m_ini           equ   11111111B           ;Default to input (high impedance).
p3m_ini           equ   00000001B           ;P33-P30 = input
                                            ;P37-P34 = output
                                            ;Digital mode for P31, P32.
                                            ;Push-pull active for P2.
p25_lo            equ   11011111B           ;bit5 = 0, Use for freq select
p25_hi            equ   00100000B           ;bit5 = 1, Use for freq select
p2m_in_all        equ   11111111B           ;
pcon_ini          equ   11111110B           ;This is the default value for PCON.
;-------------------------------------------------------------------------------

;---- INTERUPTS  LABELS --------------------------------------------------------
;  Labels/variables related to interupt control parameters.
;-------------------------------------------------------------------------------
imr_ini     equ   00000000B           ;Disable all interupt request at start up.
ipr_ini     equ   00010110B           ;Priority IRQ0,2,4,1,5,3.
irq_ini     equ   00000000B           ;Clear out all IRQ initially.
;-------------------------------------------------------------------------------

;---- TIMERs LABELS ------------------------------------------------------------
; Labels/variables related to T0 and/or T1 controls. This values are to be used with
; 4 MHz XTAL.
;-------------------------------------------------------------------------------

pre0_100usec      equ   11001001B   ;Value for PRE0 that give T0 resolution of .1msec/count
                                    ;T0 is setup as single pass only. Set bit0 for 'Modulo'.
                                    ;The 6-bit counter = 50D. Whenever this reaches 0 then
                                    ;T0 get decremented by 1. It takes .1 msec for PRE0 to
                                    ;count down from 50 to 0.
t0_1msec          equ   0AH         ;Value for T0 that would time out every 1 msec.
```

```
tmr_ini          equ   00000000B   ;Use to set initial value for TMR's register.
;=============================================================================

;==== REGISTERS DEFINITION ==================================================
;  The section below definning the labels for the particular memory location and/or group.
;=============================================================================

;===== ADDITIONAL REGISTERS ==================================================
;       These registers reside in ERF Bank F of Working Register group 0
;=============================================================================
P0       EQU     00H             ;
P1       EQU     01H             ;
P2       EQU     02H             ;
P3       EQU     03H             ;
PCON     EQU     00H             ;Port Configuration register
SMR      EQU     0BH             ;Stop Mode Recovery Register
WDTMR    EQU     0FH             ;Watch Dog Timer Register
;=============================================================================

;==== CONTROLLING INPUT REGISTERS ============================================
;  Using Working Register Group 0 for controlling input parameters such as DIP switch
;  input, mode select, switch mode and frequency select.
;=============================================================================
DIP_SW_LOWBYTE    equ   04H         ;Storing input DIP switch from 1-8.
DIP_SW_HIBYTE     equ   05H         ;Storing input DIP switch from 9-10.

DATA_IN_LOWBYTE   equ   06H         ;Storing the decoded lower 8 bit received from RF signal
DATA_IN_HIBYTE    equ   07H         ;Storing the decoded upper 2 bit received from RF signal

CONTROL_INPUTS    equ   08H         ;Store mode select, switch mode selector as defined below
                        ;  bit0  =  0 => Not : B   mode
                        ;           1 => B    mode selected.
                        ;
                        ;  bit1  =  0 => Not . C   mode.
                        ;           1 => . C    mode selected.
                        ;
                        ;  bit2  =  0 => Not : A   mode.
                        ;           1 => . A    mode selected.
                        ;
                        ;  bit3  =  Not used for ease of coding should be 0.
                        ;
                        ;  bit4  =  Not used for ease of coding should be 0.
                        ;
                        ;  bit5  =  0 => Channel 1 normal switch mode.
                        ;           1 => Channel 1 constant pressure switch mode.
                        ;
                        ;  bit6  =  0 => Channel 2 normal switch mode.
                        ;           1 => Channel 2 constant pressure switch mode.
                        ;
                        ;  bit7  =  Not use for ease of coding should be 0.
USER_FLAGS        equ   09H         ;Store user define flag control bits.
DEBOUNCE_COUNTER  equ   0AH         ;Store debounce value for reading external inputs.
P2M_COPY          equ   0BH         ;Mirror image of P2M register for reading capability.
OLD_INPUT         equ   0CH         ;Use to store previous input of the entire port.
NEW_INPUT         equ   0DH         ;Use to store current input of the entir port.
P0_INPUT          equ   0EH         ;Store P0 input.
ROTATE_COUNTER    equ   0FH         ;Store the # of rotate task to be performed.

;==== WORKING REGISTER GROUP 1 ==============================================
;  Reserve variable for Radio interupts value
;
;==== GROUP 0 ===============================================================
T0_MULTIPLIER     equ   10H         ;The number of T0 time out without Radio interupts.
                                    ;1 count = 1 msec.
T0_VALUE          equ   11H         ;Store the value of T0 when access.
BIT_COUNTER       equ   12H         ;Store the # of bits received in the data packet.
BIT_0_POSITION_LO equ   13H         ;Store the bit position of the packet data
BIT_0_POSITION_HI equ   14H         ;Store the bit position of the packet data
BIT_1_POSITION_LO equ   15H         ;Store the bit position of the packet data
BIT_1_POSITION_HI equ   16H         ;Store the bit position of the packet data
```

```
DATA_FLAGS        equ   1FH          ;Flags control bit.
                                     ;bit0 = 0 => no valid on-time pulse width detected.
                                     ;     = 1 => valid on-time pulse width detected.
                                     ;
                                     ;bit1 = 0 => first edge not detected yet.
                                     ;     = 1 => first edge already detected
                                     ;
                                     ;bit2 = 0 => 10 msec has NOT elapsed from last pulse
                                     ;     = 1 => 10 msec has elapsed from last pulse.
                                     ;
                                     ;bit7 = 0 => Ontime pulse width is for logic 1.
                                     ;     = 1 => Ontime pulse width is for logic 0
;==== GROUP 0 ========================================================================


;==== MACRO DEFINITION ===============================================================
;---------------------WATCH DOG TIMER MACROS ------------------------------------------
WDT:              MACRO
                  .byte   %5F                    ;Enable Watch dog timer.
;                 .byte   %FF                    ;Disable Watch Dog timer.
                  ENDMAC

;---------------------TIMER CONTROLS MACROS--------------------------------------------
RELOAD_T0:              MACRO
                        or      TMR,#00000001B     ;Set bit0 to reload T0, auto clear
                        ENDMAC                     ;after load by micro.

ENABLE_T0_COUNT:        MACRO
                        or      TMR,#00000010B      ;Set bit1 to enable T0 count.
                        ENDMAC

DISABLE_T0_COUNT:       MACRO
                        and     TMR,#11111101B       ;
                        ENDMAC

RELOAD_T1:              MACRO
                        or      TMR,#00000100B       ;Set bit0 to reload T1, auto clear
                        ENDMAC                       ;after load by micro.

ENABLE_T1_COUNT:        MACRO
                        or      TMR,#00001000B       ;Set bit1 to enable T1 count.
                        ENDMAC

DISABLE_T1_COUNT:       MACRO
                        and     TMR,#11110111B          ;
                        ENDMAC

RESET_MULTIPLIER:       MACRO
                        clr     T0_MULTIPLIER     ;Clear T0 multiplier.
                        ENDMAC
;---------------------TIMER CONTROLS MACROS--------------------------------------------

;---------------------DATA BITS CONTROL MACROS-----------------------------------------
RESET_BIT_COUNTER:        MACRO
                          ld     BIT_COUNTER,#01H       ;Set BIT_COUNTER to '1'(1st bit)
                          ENDMAC

RESET_BIT_0_POSITION:     MACRO
                          ld      BIT_0_POSITION_LO,#11111110B;Set BIT_POSITION to the 1st bit.
                          ld    BIT_0_POSITION_HI,#11111111B;
                          ENDMAC

RESET_BIT_1_POSITION:     MACRO
                          ld      BIT_1_POSITION_LO,#00000001B;Set BIT_POSITION to the 1st bit.
                          ld      BIT_1_POSITION_HI,#00000000B;
                          ENDMAC

LOGIC_1:                  MACRO
                          or    DATA_FLAGS,#10000000B         ;Set bit7 of flags.
                          ENDMAC
```

```
LOGIC_0:                MACRO
                        and     DATA_FLAGS,#01111111B       ;Clear bit7 of flags.
                        ENDMAC

TEN_MSEC_PASSED:        MACRO
                        or    DATA_FLAGS,#00000100B         ;Set bit2 of flags.
                        ENDMAC

TEN_MSEC_NOT_PASSED:    MACRO
                        and     DATA_FLAGS,#11111011B       ;Clear bit2 of flags.
                        ENDMAC

;----------------------DATA BITS CONTROL MACROS-----------------------------------------
   ENDMAC                           ;after load by micro.
```